NAME

astyle (Artistic Style) − prettyprint C, C++, and Java source code

SYNOPSIS

astyle [**options**] <*original-file* >*prettyprinted-file*
or
astyle [**options**] file(s)

When indenting a specific file (instead of *stdin*), the resulting indented file *retains* the original filename. The original pre-indented file is renamed, with a suffix of *.orig* added to the original filename.

By default, **astyle** is set up to indent C/C++ files, with 4 spaces per indent, a maximal indentation of 40 spaces inside continuous statements, and *no* formatting.

DESCRIPTION

astyle (Artistic Style) is a reindenter and reformatter of C, C++, and Java source code.

When indenting source code, we as programmers have a tendency to use both spaces and tab characters to create the wanted indentation. Moreover, some editors by default insert spaces instead of tabs when pressing the tab key, and other editors (Emacs for example) have the ability to *pretty up* lines by automatically setting up the white space before the code on the line, possibly inserting spaces in a code that up to now used only tabs for indentation.

Since the *number* of space characters showed on screen for each tab character in the source code changes between editors (until the user sets up the number to his liking...), one of the standard problems facing programmers when moving from one source code editor to another is that code containing both spaces and tabs that was up to now perfectly indented, suddenly becomes a mess to look at when changing to another editor. Even if you as a programmer take care to *only* use spaces or tabs, looking at other people's source code can still be problematic.

To address this problem I have created **astyle** (Artistic Style) — a series of filters, written in C++, that automatically reindent & reformat C/C++/Java source files. These can be used from a command line, or they can be incorporated as classes in another C++ program.

OPTIONS

Long options (starting with −−) must be written one at a time. Short options (starting with −) may be appended together. Thus, **−bps4** is the same as **−b −p −s4.**

−a
−−brackets=*attach*

Attach brackets to pre-block code (i.e., Java/K&R style).

Sample formatting with this option:

```
if (isFoo){
    bar();
} else {
    anotherBar();
}
```

−B
−−indent-brackets

Add extra indentation to *{* and *}* block brackets.

Sample formatting without this option:

```
if (isFoo)
{
    bar();
}
else
{
    anotherBar();
}
```

Sample formatting with this option:

```
if (isFoo)
    {
    bar();
    }
else
    {
    anotherBar();
    }
```

−b
−−brackets=*break*

Break brackets from pre-block code (i.e., ANSI C/C++ style).

Sample formatting with this option:

```
if (isFoo)
{
    bar();
}
else
{
    anotherBar();
}
```

−C
−−indent-classes

Indent *class* blocks, so that the inner *public:*, *protected:*, and *private:* headers are indented in relation to the *class* block.

Sample formatting without this option:

```
class Foo
{
public:
    Foo();
    virtual ˜Foo();
};
```

Sample formatting with this option:

```
class Foo
{
    public:
      Foo();
      virtual ˜Foo();
};
```

−c
−−mode=*c*

Indent a C or C++ source file (default).

−E
−−fill-empty-lines

Fill empty lines with the white space of their previous lines.

−G
−−indent-blocks

Add extra indentation for entire blocks (including brackets).

Sample formatting without this option:

```
if (isFoo)
{
    bar();
}
else
    anotherBar();
```

Sample formatting with this option:

```
if (isFoo)
    {
      bar();
    }
else
    anotherBar();
```

−h
−?
−−help
Print help message on *stderr*.

−j
−−mode=*java*
Indent a Java(TM) source file.

−K
−−indent-cases
Indent *case XXX:* lines, so that they are flush with their bodies.

Sample formatting without this option:

```
switch (foo)
{
case 1:
    {
      a += 2;
      break;
    }
default:
    {
      a += 2;
      break;
    }
}
```

Sample formatting with this option:

```
switch (foo)
{
    case 1:
    {
      a += 2;
      break;
    }
    default:
    {
      a += 2;
      break;
    }
}
```

−L

−−indent-labels

Indent labels so that they appear one indent less than the current indentation level, rather than being flushed completely to the left (which is the default).

Sample formatting without this option:

```
int foospace()
{
    while (isFoo)
    {
      ...
      goto error;
error:
      ...
    }
}
```

Sample formatting with this option:

```
int foospace()
{
    while (isFoo)
    {
      ...
      goto error;
    error:
      ...
    }
}
```

−l
−−brackets=*linux*

Break definition-block brackets and attach command-block brackets.

Sample formatting with this option:

```
namespace foospace
{
    int Foo()
    {
      if (isBar) {
          bar();
          return 1;
      } else
          return 0;
    }
}
```

−M#
−−max-instatement-indent=*#*

Indent a maximal # spaces in a continuous statement, relatively to the previous line.

−m
−−min-conditional-indent=*#*

Indent a minimal # spaces in a continuous conditional belonging to a conditional header.

Sample formatting without this option:

```
// default setting makes this non-bracketed code clear
if (a < b
      || c > d)
    foo++;
// but creates an exaggerated indent in this bracketed code
```

```
if (a < b
      || c > d)
{
    foo++;
}
```

Sample formatting with **−−min-conditional-indent**=*0:*

```
// setting makes this non-bracketed code less clear
if (a < b
    || c > d)
    foo++;
// but makes this bracketed code prettier
if (a < b
    || c > d)
{
    foo++;
}
```

−N
−−indent-namespaces

Indent the contents of namespace blocks.

Sample formatting without this option:

```
namespace foospace
{
class Foo
{
    public:
      Foo();
      virtual ˜Foo();
};
}
```

Sample formatting with this option:

```
namespace foospace
{
    class Foo
    {
      public:
          Foo();
          virtual ˜Foo();
    };
}
```

−O
−−one-line=*keep-blocks*

Don’t break blocks residing completely on one line.

−o
−−one-line=*keep-statements*

Don’t break lines containing multiple statements into multiple single-statement lines.

−P
−−pad=*all*

Insert space paddings around operators *and* parentheses.

Sample formatting without this option:

```
if (isFoo)
    a = bar((b-c)*a,*d--);
```

Sample formatting with this option:

```
if ( isFoo )
    a = bar( ( b - c ) * a, *d-- );
```

−p
−−pad=*oper*

Insert space paddings around operators only.

Sample formatting without this option:

```
if (isFoo)
    a = bar((b-c)*a,*d--);
```

Sample formatting with this option:

```
if (isFoo)
    a = bar((b - c) * a, *d--);
```

−−pad=*paren*

Insert space paddings around parentheses only.

Sample formatting without this option:

```
if (isFoo)
    a = bar((b-c)*a,*d--);
```

Sample formatting with this option:

```
if ( isFoo )
    a = bar( ( b-c )*a, *d-- );
```

−S
−−indent-switches

Indent *switch* blocks, so that the inner *case XXX:* headers are indented in relation to the switch block.

Sample formatting without this option:

```
switch (foo)
{
case 1:
    a += 2;
    break;
default:
    a += 2;
    break;
}
```

Sample formatting with this option:

```
switch (foo)
{
    case 1:
      a += 2;
      break;
    default:
      a += 2;
      break;
}
```

−s
−s#
−−indent=*spaces=#*

Indent using # spaces per indent. Not specifying # will result in a default of 4 spaces per indent.

−−style=*ansi*
−−style=*kr*

−−style=*gnu*
−−style=*java*
−−style=*linux*

Choose ANSI, Kernighan & Ritchie, GNU, Java, or GNU/Linux style prettyprinting.

−−suffix=*####*

Append the suffix *####* instead of *.orig* to the original filenames.

−t
−t#
−−indent=tab=#

Indent using tab characters, assuming that each tab is # spaces long. Not specifying # will result in a default assumption of 4 spaces per tab.

−v
−−version

Print version number on *stderr*.

−X
−−errors-to-standard-output

Print errors and help information to *stdout*, rather than to *stderr*.

DEFAULT OPTIONS

astyle looks for a default options file in the following order:

1. The contents of the file named by the **ARTISTIC_STYLE_OPTIONS** environment variable, if that variable exists.
2. The file called *.astylerc* in the directory pointed to by the **HOME** environment variable (i.e., *$HOME/.astylerc*).
3. The file called *.astylerc* in the directory pointed to by the **HOMEPATH** environment variable (i.e., *$HOMEPATH/.astylerc*).

At most *one* default options file is processed, and if one is found, the options in it will be parsed *before* the command-line options.

Options within the default option file may be written without the preliminary `−` or `−−`.

Options may be set apart by newlines, tabs or spaces.

Long options may be written in the options file without the preceding `−−`.

Lines within the options file that begin with # are considered comments.

Here is a sample default options file:

```
# default parsing is of java files
mode=java
# brackets should be attached to pre-bracket lines
brackets=attach
# set 6 spaces per indent
indent=spaces=6
# indent switch blocks
indent-switches
# suffix of original files should be .pre
suffix=.pre
```

STYLE SAMPLES

Here are samples of each of the supported formatting styles:

−−style=*ansi*

ANSI style formatting/indenting.

```
namespace foospace
{
    int Foo()
    {
        if (isBar)
        {
            bar();
            return 1;
        }
        else
            return 0;
    }
}
```

−−style=*kr*

Kernighan & Ritchie style formatting/indenting.

```
namespace foospace {
    int Foo() {
        if (isBar) {
            bar();
            return 1;
        } else
            return 0;
    }
}
```

−−style=*linux*

GNU/Linux style formatting/indenting (brackets are broken apart from class/function declarations, but connected to command lines, and indents are set to 8 spaces).

```
namespace foospace
{
        int Foo()
        {
                if (isBar) {
                        bar();
                        return 1;
                } else
                        return 0;
        }
}
```

−−style=*gnu*

GNU style formatting/indenting.

```
namespace foospace
  {
    int Foo()
      {
        if (isBar)
          {
            bar();
            return 1;
          }
        else
          return 0;
      }
}
```

−−style=*java*

Java style formatting/indenting.

```
class foospace {
    int Foo() {
        if (isBar) {
            bar();
            return 1;
        } else
            return 0;
    }
}
```

ENVIRONMENT VARIABLES

ARTISTIC_STYLE_OPTIONS	Primary default option filename.
HOME	Secondary default option file directory (*$HOME/.astylerc*).
HOMEPATH	Tertiary default option file directory (*$HOMEPATH/.astylerc*).

SEE ALSO

awkpretty(1), **html-pretty**(1), **indent**(1), **pindent**(1), **pretty**(1), **sf3pretty**(1), **tex-pretty**(1).

AUTHOR

Tal Davidson
Israel
email: `davidsont@bigfoot.com`
WWW URL: `http://astyle.sourceforge.net/astyle`